Lawrence Berkeley National Laboratory
Recent Work

Title
Referential Integrity Revisited: An Object-Oriented Perspective

Permalink
https://escholarship.org/uc/item/5wn8t1cq

Author
Markowitz, V.M.

Publication Date
1990-06-01
Referential Integrity Revisited: An Object-Oriented Perspective

V.M. Markowitz

June 1990

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.
DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.
REFERENTIAL INTEGRITY REVISITED:
AN OBJECT-ORIENTED PERSPECTIVE

Victor M. Markowitz

Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, California 94720

June 1990

Proceedings of the 16th International Conference on Very Large Data Bases (VLDB), August 13-16, 1990, Brisbane, Australia

This work was supported by the Office of Health and Environmental Research Program and the Applied Mathematical Sciences Research Program, of the Office of Energy Research, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.
REFERENTIAL INTEGRITY REVISITED: AN OBJECT-ORIENTED PERSPECTIVE

VICTOR M. MARKOWITZ

Lawrence Berkeley Laboratory
Information and Computing Sciences Division
Computer Science Research and Development Department
1 Cyclotron Road, Berkeley, CA 94720

ABSTRACT
Referential integrity underlies the relational representation of object-oriented structures. The concept of referential integrity in relational databases is hindered by the confusion surrounding both the concept itself and its implementation by relational database management systems (RDBMS). Most of this confusion is caused by the diversity of relational representations for object-oriented structures. We examine the relationship between these representations and the structure of referential integrity constraints, and show that the controversial structures either do not occur or can be avoided in the relational representations of object-oriented structures.

Referential integrity is not supported uniformly by RDBMS products. Thus, referential integrity constraints can be specified in some RDBMSs non-procedurally (declaratively), while in other RDBMSs they must be specified procedurally. Moreover, some RDBMSs do not allow the specification of certain referential integrity constraints. We discuss the referential integrity capabilities provided by three representative RDBMSs, DB2, SYBASE, and INGRES.

I. INTRODUCTION
The database design process involves specifying the objects and object connections relevant to the database application. In relational databases objects are represented by relation tuples, while object connections are represented by references between tuples. Such references are enforced in relational databases by referential integrity constraints [2]. There are two approaches to the specification of these constraints in relational databases. In the Universal Relation (UR) approach [11], referential integrity constraints are implied by associating different relations with common attributes; the referential integrity meaning of relations sharing common attributes is defined by a set of rules, called UR assumptions. The UR assumptions make the description of object structures extremely difficult and not entirely reliable, mainly because they require excessively complex attribute name assignments.

A different approach to the specification of referential integrity constraints is to associate explicitly a foreign-key in one relation with the primary-key of another relation [5]. Such constraints are a special case of inclusion dependencies [4]. Every explicit referential integrity constraint is usually associated with a referential integrity rule which defines the behavior of the relations involved in the constraint under insertion, deletion, and update. Explicit referential integrity constraints are easier to specify and understand than the implicit referential integrity constraints of the UR approach, because they are closer to the way users describe object structures. However, the referential integrity concept is still surrounded by confusion, as illustrated by the successive modifications of the original definition of [2] (e.g. see [3], [5]). Thus, certain referential integrity structures have unclear semantics, and therefore must be 'treated with caution' [6]. Obviously, a non-technical user cannot be expected to manage the complexities of such a task.

In this paper we examine the characteristics of referential integrity constraints involved in the relational representation of object-oriented structures. We show that the controversial structures discussed in [6] can be avoided without any effect on the capability of relational schemas to represent object structures. We explore the characteristics of referential integrity constraints in the context of relational schemas representing Extended Entity-Relationship (EER) object structures. We have selected the EER model because of its widespread use in designing relational databases [19]. However, our results apply to any object-oriented data model that supports generalization and aggregation [9].

We have shown in [15] that an EER schema can be represented by a Boyce-Codd Normal Form (BCNF) relational schema of the form \((R, F \cup I)\), where \(R\) denotes a set of relation-schemes, and \(F\) and \(I\) denote...
sets of key and inclusion dependencies, respectively. Informally, relation-schemes represent object-sets, and inclusion dependencies represent the existence dependencies inherent to object connections. The inclusion dependencies in these schemas are key-based, that is, are referential integrity constraints, and relation-schemes correspond to either unique or multiple (embedded) object-sets. In [12] we have shown that the mapping process involved in representing EER schemas by relational schemas, can be expressed as the composition of (i) the mapping of EER schemas into a relational schemas, where every relation-scheme corresponds to a unique EER object-set, followed by (ii) relation-scheme mergings, that result in relation-schemes representing multiple object-sets.

We examine the structure of referential integrity constraints involved in relational schemas whose relation-schemes represent unique EER object-sets. We show that in such schemas referential integrity constraints can be associated with one out of four possible referential integrity rules. Next, we examine the effect of merging on the structure of referential integrity constraints, and show that merging entails associating some referential integrity constraints with an additional, fifth, referential integrity rule. In contrast, seven referential integrity rules are defined in [3] and [5]; we show that the two extra rules are not needed for representing EER object structures.

Currently, several relational database management systems (RDBMS), notably IBM's DB2, SYBASE, and INGRES, provide support for referential integrity. Interestingly, these systems provide different capabilities for specifying referential integrity constraints. Thus, DB2 [7] allows non-procedural (declarative) specifications of referential integrity constraints, but with certain restrictions on the allowed structures. Conversely, SYBASE [18] and INGRES [8] do not support declarative specifications of referential integrity constraints, and provide instead mechanisms (triggers in SYBASE and rules in INGRES) for specifying such constraints procedurally. We discuss in this paper the referential integrity capabilities of DB2, SYBASE, and INGRES. We show that some of the restrictions imposed by DB2 are too stringent. We compare the SYBASE and INGRES mechanisms for specifying referential integrity constraints, and discuss their limitations.

The paper is organized as follows. In section 2 we briefly review the relational and EER concepts used in this paper, and the representation of EER schemas. In section 3 we examine two controversial foreign-key structures in the context of relational schemas representing EER object structures. The semantics of referential integrity rules in the context of relational schemas representing EER schemas, is explored in section 4. In section 5 we examine the effect of merging relations on the structure of referential integrity constraints. The referential integrity capabilities of DB2, SYBASE, and INGRES are examined in section 6. We conclude with a summary. The procedures for mapping EER schemas into relational schemas, and for merging relation-schemes in relational schemas are given in the appendix.

II. PRELIMINARY DEFINITIONS

In this section we review briefly the relational and Extended Entity-Relationship (EER) concepts used in this paper, and the representation of EER object structures using relational constructs.

2.1 Relational Concepts.

We use letters from the beginning of the alphabet to denote attributes and letters from the end of the alphabet to denote sets of attributes. We denote by t a tuple and by t[|W|] the sub-tuple of t corresponding to the attributes of W.

A relational schema is a pair (\(R, \Delta\)), where R is a set of relation-schemes and \(\Delta\) is a set of dependencies over R. We consider relational schemas with \(\Delta = F \cup I\), where F and I denote sets of functional and inclusion dependencies, respectively. A relation-scheme is a named set of attributes, R\(i(X_i)\), where R\(i\) is the relation-schema name and \(X_i\) denotes the set of attributes. Every attribute is assigned a domain, and every relation-scheme, R\(i(X_i)\), is assigned a relation (value), r\(i\). The projection of such a relation, r\(i\), on a subset of \(X_i\), W, is denoted \(\pi_W(r_i)\), and is equal to \(\{t[|W|] | t \in r_i\}\). Two attributes are said to be compatible if they are associated with the same domain, and attribute sets X and Y are said to be compatible iff there exists a one-to-one correspondence of compatible attributes between X and Y.

Let R\(i(X_i)\) be a relation-scheme associated with relation r\(i\). A functional dependency over R\(i\) is a statement of the form R\(i\): \(Y \rightarrow Z\), where \(Y\) and \(Z\) are subsets of \(X_i\); R\(i\): \(Y \rightarrow Z\) is satisfied by r\(i\) iff for any two tuples of r\(i\), t and t', t[|Y|] = t'[|Y|] implies t[|Z|] = t'[|Z|]. A key associated with R\(i\) is a subset of \(\bigcup X_i\), \(K_i\), such that R\(i\): \(K_i \rightarrow X_i\) is satisfied by any r\(i\) associated with R\(i\) and there does not exist any proper subset of \(K_i\) having this property. A relation-scheme can be associated with several candidate keys from which one primary-key is chosen.

Let R\(i(X_i)\) and R\(j(X_j)\) be two relation-schemes associated with relations r\(i\) and r\(j\), respectively. An
inclusion dependency is a statement of the form \( R_i[Y] \subseteq R_j[Z] \), where \( Y \) and \( Z \) are compatible subsets of \( X_i \) and \( X_j \), respectively; \( R_i[Y] \subseteq R_j[Z] \) is satisfied by \( r_i \) and \( r_j \) iff \( \pi_Y(r_i) \subseteq \pi_Z(r_j) \). If \( Z \) is the primary-key of \( R_j \) then \( R_i[Y] \subseteq R_j[Z] \) is said to be key-based, and \( Y \) is called a foreign-key of \( R_i \).

Key-based inclusion dependencies are referential integrity constraints ([2], [5]).

2.2 The Extended Entity-Relationship Model.

The basic concepts of the Entity-Relationship model, (entity, relationship, attribute, entity-set, relationship-set, value-set, entity-identifier, weak entity-set, relationship cardinality, role) have been repeatedly reviewed (e.g. [19]) since their original definition in [1]. We refer commonly to entities and relationships as objects. The Extended Entity-Relationship (EER) model considered in this paper has two additional abstraction mechanisms, generalization and full aggregation. Generalization ([9], [19]) is an abstraction mechanism that allows viewing a set of entity-sets as a single generic entity-set. The inverse of generalization is called specialization. The full capability of aggregation is provided in the EER model by allowing relationship-sets to associate any object-set, rather than only entity-sets.

An EER schema can be represented as an acyclic directed graph, called EER diagram: entity-sets, relationship-sets, and attributes, are represented by rectangle, diamond, and ellipse shaped vertices, respectively; and the connection of EER object-sets and attributes is represented by directed edges. Thus, there are directed edges: from relationship-sets to the object-sets they associate, labeled by a cardinality (1 (one) or M (many)); from weak entity-sets to the entity-sets on which they depend for identification, labeled ID; from specialization entity-sets to generic entity-sets, labeled ISA; and from object-sets to their attributes. The acyclicity of EER diagrams is implied by certain restrictions satisfied by EER schemas ([9], [15]). A self-explanatory EER diagram example is shown in figure 1.

2.3 Relational Representation of Extended Entity-Relationship Schemas.

Relational schemas representing EER object structures are of the form \((R, F \cup I)\), where \(R\), \(F\), and \(I\) denote sets of relation-schemes, functional dependencies, and inclusion dependencies, respectively [15]. Informally, relation-schemes represent EER object-sets, inclusion dependencies represent the existence dependencies inherent to object-set connections, and functional dependencies represent entity-identifiers and relationship cardinalities. In [15] we have shown that an EER schema can be represented by a BCNF relational schema, such that every relation-scheme corresponds to a unique object-set, functional dependencies are key dependencies, and inclusion dependencies are key-based, that is, are referential integrity constraints.

A procedure for mapping EER schemas into relational schemas based on the algorithms developed in [15], called \(Rmap\), is given in the appendix, and is exemplified in figure 2. Concerning the assignment of names to relational attributes, various techniques can be employed [16]. In order to keep \(Rmap\) independent of a specific name assignment for relational attributes, we use only symbolic names for attributes (e.g. \(A_{42}\) represents the second attribute of the fourth relation).

![Fig.1 An Extended Entity-Relationship Schema.](image)
III. PARTLY NULL AND OVERLAPPING KEYS

In [6] Date discusses the problems caused by overlapping \(1\) (foreign and primary) keys, and partly null \(2\) foreign-keys, and points out their obscure semantics. We show below that in relational schemas representing EER object structures overlapping keys do not occur, and partly null foreign-keys can be avoided.

3.1 Overlapping Keys.

As mentioned in the previous section, the relational schema representation of an EER object structure is of the form \((R, F \cup I)\), where \(R, F\), and \(I\) denote sets of relation-schemes, functional dependencies, and inclusion dependencies, respectively. The following proposition shows that relational schemas generated by \(R_{map}\) do not involve overlapping keys.

**Proposition 1.** Let \(RS = (R, F \cup I)\) be a relational schema generated by \(R_{map}\). Let \(R_i(X_i)\) be a relationscheme of \(R\), and let \(FK_i\) denote the union of all foreign-keys, \(FK_i\), associated with \(R_i\). Then every foreign-key \(FK_i\) associated with \(R_i\) satisfies the following conditions: \(FK_i\) is either included in, or disjoint with, the primary-key of \(R_i\); and \(FK_i\) is either equal to, or disjoint with, every other foreign-key of \(R_i\).

**Proof:** see proposition 4.1 of [14].

3.2 Partly Null Foreign-Keys.

We examine below the EER modeling of missing information, and then show how partly null foreign-keys can be avoided in relational schemas representing EER object structures.

The EER modeling of missing information involves allowing attributes to have unknown or inapplicable values, and allowing partially specified relationships, that is, allowing objects involved in relationships to be unknown or inapplicable. Consider the EER schema of figure 1, where entity-set FACULTY is associated with attribute RANK; the value of RANK can be unknown for some faculty members. If RANK, however, is associated with entity-set PERSON, then for a person who is not a faculty member, the RANK value is inapplicable.

Partially specified relationships are needed for representing embedded real-world associations. For example, suppose that a ternary relationship-set \(ASSIGNED\) involves entity-sets FACULTY, DEPARTMENT, and COURSE, and represents the assignment of faculty members to courses offered by departments; then in \(ASSIGNED\) relationships representing courses that are offered by departments, but that are not assigned to a faculty member, the FACULTY entity is unknown. Inapplicable attribute values and partially specified relationships can be avoided as follows:

- if an attribute \(A\) associated with entity-set \(E\) has inapplicable values for some entities of \(E\), then \(A\) can be associated with a (possibly newly defined) specialization of \(E, E^\prime\), so that \(A\) is always applicable for the entities of \(E^\prime\); for example, if attribute RANK above is associated with FACULTY (as shown in figure 1), then RANK is always applicable.

- if a relationship-set includes partially specified relationships, then it can be decomposed into independent or related (by aggregation) relationship-sets involving only fully specified relationships (note that binary relationship-sets consist only of fully specified relationships); for example, relationship-set \(ASSIGNED\) above can be decomposed into relationship-sets OFFER and TEACH (as shown in figure 1) that involve only fully specified relationships.

Consequently, only unknown attribute values are needed for the EER modeling of missing information, while partially specified relationships and inapplicable attribute values can be avoided. Note that for convenience entity-identifier attributes are usually not allowed to have unknown values.

Relational modeling of missing information employs special purpose null values. The various meanings associated with nulls are generally compressed into two [3]: inapplicable values and unknown (but applicable) values. From the discussion concerning the EER modeling of missing information, it follows that nulls representing inapplicable values can be avoided in databases associated with schemas representing EER object structures. Nulls representing unknown values, however, can represent in such databases either an unknown EER attribute value or an unknown (relational) foreign-key attribute value. In relational schemas such as those generated by \(R_{map}\), in which every relation-scheme corresponds to a unique object-set, primary-keys and foreign-keys are not allowed to have null values; these constraints represent the restrictions of not allowing object-identifiers to have unknown values, and not allowing relationships to be partially specified. Note that for foreign-keys these constraints are stronger than the referential-integrity constraint specified in [2]. However, in [3] Codd states that nulls should not be allowed for foreign-keys in most cases, but only when

\[\text{null}
\]

\[\text{inapplicable}
\]

\[\text{unknown}
\]

\[\text{values}
\]

\[\text{foreign}
\]

\[\text{primary}
\]

\[\text{keys}
\]

\[\text{unknown}
\]

\[\text{values}
\]

\[\text{inapplicable}
\]

\[\text{values}
\]

\[\text{unknown}
\]

\[\text{values}
\]

\[\text{null}
\]

\[\text{values}
\]

\[\text{foreign}
\]

\[\text{primary}
\]

\[\text{keys}
\]
Chapter 4: Referential Integrity Rules

4.1 Existence Dependency Types

Object-oriented structures imply certain existence dependencies that must be satisfied by updates. In an object-oriented environment an elementary update consists of modifying an attribute value, removing an object from an object-set, or adding a new object to an object-set. In order to satisfy the existence dependencies, usually an object \( x \) that is existence dependent on another object \( y \), cannot be added before \( y \) is added, and \( y \) cannot be removed before \( x \) is removed. Intuitively, \( y \) blocks the addition of \( x \), and \( x \) blocks the removal of \( y \); therefore, the existence dependency of \( x \) and \( y \) is said to be of type block. We propose an additional, type of existence dependency, called trigger: if an object \( x \) is trigger existence dependent on another object \( y \), then the removal of \( y \) triggers the removal of \( x \) (instead of \( x \) blocking the removal of \( y \)). Note that existence dependencies between objects do not affect attribute modifications, since such modifications are local to the objects.

The two types of existence dependency above are specified for pairs of object-sets, rather than pairs of individual objects, and can be represented in EER diagrams as follows: the block existence dependency is considered the default type, and therefore is not explicitly represented; the existence dependency of type trigger is represented by associating the edges connecting the corresponding object-sets with a 'v' label (see figure 1).

4.2 Referential Integrity Rules

Let \( R_i(X_i) \) and \( R_j(X_j) \) be two relation-schemes associated with relations \( r_i \) and \( r_j \), respectively. A referential integrity constraint \( R_i[Y] \subseteq R_j[K] \) is associated with a referential integrity rule consisting of an insert-rule, a delete-rule and an update-rule [5]. There is a unique insert-rule, restricted, which asserts that inserting a tuple \( t \) into \( r_i \) can be performed only if the tuple of \( r_j \) referenced by \( t \) already exists. The delete and update rules define the effect of deleting (resp. updating the primary-key value in) a tuple \( t' \) of

\[ r_j: \text{a restricted delete (resp. update) rule asserts that the deletion (resp. update) of } t' \text{ cannot be performed if there exist tuples in } r_i \text{ referencing } t'; \text{ a cascades delete (resp. update) rule asserts that the deletion (resp. update) of } t' \text{ implies deleting (resp. updating the sub-tuple } t'[Y] \text{ in } \text{the tuples of } r_i \text{ referencing } t'; \text{ and a nullifies delete (resp. update) rule asserts that the deletion (resp. update) of } t' \text{ implies setting to null the sub-tuple } t'[Y] \text{ in all the tuples } t \text{ of } r_i \text{ referencing } t'. \]

Procedures mapping EER schemas into relational schemas, such as Rmap, usually assume that existence dependencies are of type block. The semantics of such existence dependencies is expressed by associating the corresponding referential integrity constraints with restricted insert and delete rules. The new type of existence dependency introduced above, entails associating referential integrity constraints corresponding to trigger existence dependencies with cascades delete-rules. Finally, the preservation of existence dependencies under attribute modifications in object-oriented environments, is expressed by associating referential integrity constraints with cascades update-rules. Thus, the referential integrity constraints in figure 2, for example, must be associated with restricted insert-rules and cascades update-rules; the delete-rules are restricted for all the constraints with the exception of \( R_9[A_9] \subseteq R_6[A_4] \) and \( R_6[A_9] \subseteq R_7[A_7] \), for which the delete-rules are cascades.

The mapping of EER schemas into relational schemas can be straightforwardly extended with the explicit generation of referential integrity rules. Thus, procedure Rmap is extended by associating every referential integrity constraint ref with a restricted insert-rule and a cascades update-rule; the delete-rule depends on the type of existence dependency corresponding to ref; if the type of the existence dependency is block then ref is associated with a restricted delete-rule, otherwise (if the type is trigger) ref is associated with a cascades delete-rule.

4.3 Conflicting Existence Dependencies

Objects can block or trigger the removal of other objects with which they are involved in existence dependencies not only directly, but also transitively. Thus, an object \( y \) can trigger the removal of an object \( x \) if \( y \) can trigger the removal of an object \( z \) on which \( x \) is trigger existence dependent; conversely, if \( x \) is block existent dependent on \( z \) then \( x \) blocks the removal of \( y \). For example, suppose that in the EER schema of figure 1 the existence dependencies of INSTRUCTOR on STUDENT, STUDENT on PERSON, and FACULTY on PERSON, are of type trigger, while the existence dependency of INSTRUCTOR on FACULTY is of...
type block; then an INSTRUCTOR entity blocks the removal of a PERSON entity (via FACULTY), while a PERSON entity can trigger the removal of an INSTRUCTOR entity (via STUDENT).

If an object $x$ blocks the removal of an object $y$ which, in turn, can trigger the removal of $x$, then the result of removing $y$ is unpredictable. In the example above, for instance, removing a PERSON entity depends on the order in which the removal is performed (i.e., first via FACULTY, or first via STUDENT). Consequently, objects should not be allowed to block the removal of objects that can trigger their removal. In terms of the EER object structure, this restriction can be stated as follows:

Let $O_i$ be an object-set of an EER schema $ES$, and let $\text{Trig}(O_i)$ be the set of object-sets of $ES$ consisting of all the object-sets that are trigger existent dependent (directly or transitively) on $O_i$. Then, for every $O_i$ of $ES$, object-sets of $\text{Trig}(O_i)$ are not allowed to be (directly) block existent dependent either on $O_i$ or on other object-sets of $\text{Trig}(O_i)$.

In the example above, for instance, $\text{Trig}($PERSON$)$ consists of FACULTY, STUDENT, and INSTRUCTOR, therefore the existence dependency of INSTRUCTOR on FACULTY cannot be of type block.

The following proposition defines the effect of the restriction above on the referential integrity rules involved in relational schemas representing EER object structures.

**Proposition 2.** Let $RS = (R, F) \cup I$ be a relational schema generated by $Rmap$. Let $G_l = (V, H)$ be the referential integrity graph associated with $RS$, defined as follows: $V = R$, and $H = (R_i \rightarrow R_j | R_i \subseteq R_j \subseteq I \subseteq I)$. Let $\text{Case}(R_l)$ be the subset of $R$ consisting of all the vertices that are connected to $R_l$ in $G_l$ by directed paths consisting of edges that correspond to referential integrity constraints associated with $\text{cascades}$ delete-rules. Then for every $R_l$ of $R$, the referential integrity constraints corresponding to the edges of $G_l$ that connect vertices of $\text{Case}(R_l)$ with $R_l$ or other vertices of $\text{Case}(R_l)$, are not allowed to be associated with restricted delete-rules.

**Proof Sketch.** $Rmap$ generates referential integrity constraints associated with a referential integrity graph, $G_l$, that is isomorphic to a subgraph of the corresponding EER diagram (proposition 4.1 in [14]). The condition of the proposition follows from the restriction above specified for EER schemas.

---

1 In EER diagram terms, every object-set of $\text{Trig}(O_i)$ is connected to $O_i$ by a directed path consisting of $\triangledown$-labeled edges.

V. THE EFFECT OF MERGING ON REFERENTIAL INTEGRITY CONSTRAINTS

In this section we examine the effect of merging relations on the structure of foreign-keys and referential integrity constraints.

Merging brings about the need to allow certain foreign-key attributes to have null values. We have shown in [12] that merging relations requires the introduction of additional null constraints [10] for restricting the way in which null values appear in merged relations. A procedure for merging relations in relational databases that preserves the information-capacity and the normal form of the corresponding schemas, has been proposed in [12]. The merging procedure developed in [12] may generate inclusion dependencies that are not key-based, that is, are not referential integrity constraints. We consider below a merging procedure that generates only key-based inclusion dependencies and involve only simple null constraints, that indicate the attributes that are not allowed to have null values.

A restricted version of the procedure proposed in [12] for merging relation-schemes in relational schemas, called $Rmerge$, is given in the appendix. Given a schema $RS = (R, F \cup I)$ generated by $Rmap$, and a subset $\bar{R}$ of $R$, such that the primary-keys associated with the relation-schemes of $\bar{R}$ are pairwise compatible, $Rmerge$ maps $RS$ into a new relational schema, $RS' = (R', F' \cup I')$, where $R'$ results by replacing the relation-schemes of $\bar{R}$ with a new relation-scheme, $R_m$, and $F'$ and $I'$ consist of adjusted key and inclusion dependencies, respectively. Merging is achieved by outer-joining the relations associated with the relation-schemes of $\bar{R}$, so that the relation associated with $R_m$, $r_m$, includes tuples corresponding to every object represented in the merged relations. The dependencies associated with $R_m$ ensure that the relations involved in merging can be reconstructed from $r_m$, so that the schema generated by $Rmerge$, $RS'$, has equivalent information-capacity with $RS$. An example of merging is shown in figure 3, where $Rmerge$ is applied on the relational schema of figure 4 in order to merge relation-schemes $R_7, R_8$ and $R_9$ into $R'$.

Following merging, the referential integrity constraints involving the relation-schemes of $\bar{R}$ are replaced with referential integrity constraints involving the new relation-scheme, $R_m$, and some of the foreign-keys associated with $R_m$ are allowed to have null values. Clearly, the referential integrity constraints involving $R_m$ must be associated with restricted insert-rules and $\text{cascades}$ update-rules, like the corresponding referential integrity constraints involving relation-schemes of $\bar{R}$. Concerning the delete-rules,
**Rmerge** assumes that the referential integrity constraints involving relation-schemes of \( R \) are associated with restricted delete-rules, and therefore the referential integrity constraints involving \( R_m \) are also associated with restricted delete-rules. If cascades delete-rules are also considered, then **Rmerge** must be extended as follows:

- if a referential integrity constraint involving \( R_m \), ref, is derived from a referential integrity constraint associated with a cascades delete-rule, then ref is associated either with (a) a nullifies delete-rule, if ref involves a foreign-key of \( R_m \) that is allowed to have null values, or (b) a cascades delete-rule, otherwise.

For example, referential integrity constraint \( R_7[A_81 \subseteq R_4[A_4] \) in the relational schema of figure 3, is associated with a nullifies delete-rule (because it corresponds to a referential integrity constraint associated with a cascades delete-rule and \( A_8 \) is allowed to have null values), while the other referential integrity constraints involving relation-scheme \( R_7 \) in this schema are associated with restricted delete-rules.

The effect of merging on the structure of foreign-keys and referential integrity constraints is captured by the following proposition.

**Proposition 3.** Let \( RS = (R, F \cup I) \) be a relational schema generated by **Rmap**, and let \( RS^* = (R', F' \cup I') \) be the result of applying **Rmerge** on \( RS \). Then (a) \( RS^* \) satisfies the conditions of propositions 1 and 2. (b) In every relation-scheme \( R'[X'] \) of \( R' \), if a foreign-key \( FK_{i} \), is allowed to have null values then \( FK_{i} \) consists of a single attribute that does not appear in any other foreign or primary key of \( R' \). (c) If there exists a directed cycle in the referential integrity graph \( G_i \) associated with \( RS^* \), then at least one of the referential integrity constraints corresponding to the edges of this cycle:

(i) involves a foreign-key that is allowed to have null values, and (ii) is associated with either a restricted or a nullifies delete-rule.

**Proof Sketch.** (a) The proof follows the specification of **Rmerge.** (b) See proposition 4.2 in [14]. (c) **Rmap** generates referential integrity constraints associated with a referential integrity graph, \( G_i \), that is isomorphic to a subgraph of the corresponding EER diagram. Since EER diagrams are acyclic (see [15]) \( G_i \) is also acyclic. Cycles in \( G_i \), may result from merging relation-schemes in \( RS \), and the proof is based on the conditions of proposition 1 and the specification of **Rmerge.**

The proposition above shows that merging does not alter the properties of propositions 1 and 2. Thus, relational schemas undergoing merging are still free of the undesirable foreign-key and referential integrity structures discussed in sections 3 and 4. It can be verified that proposition 3 is valid not only for the restricted merging procedure considered above, but also for extended merging procedures such as that defined in [12].

Several remarks concerning the referential integrity rules involved in the relational representation of object-oriented structures, are in order. Only restricted insert-rules, restricted delete-rules, and cascades update-rules are required for representing existence dependencies of type block between object-sets. Cascades delete-rules are required for representing existence dependencies of type trigger, and nullifies delete-rules are required only if relations are allowed to represent multiple object-sets (e.g. following merging). The referential integrity constraints involved in relational schemas representing object-oriented structures are always associated with cascades update-rules, therefore nullifies and restrict update-rules can be discarded. Cascades update-rules become superfluous if updates of primary-key attributes are not allowed. While such a restriction is unreasonable for regular relational attributes, this restriction underlies the definition of surrogate attributes [2]. Consequently, if surrogate attributes are used as primary and foreign key attributes, then cascades update-rules are not necessary.

### Relation-Scheme, Primary Key, Referential Integrity Constraints

<table>
<thead>
<tr>
<th>Relation-Scheme</th>
<th>Primary Key</th>
<th>Referential Integrity Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1(A_{11}) )</td>
<td>( A_{11} )</td>
<td>( R_4[A_{4}] \subseteq R_1[A_{11}] )</td>
</tr>
<tr>
<td>( R_2(A_{21}) )</td>
<td>( A_{21} )</td>
<td>( R_5[A_{5}] \subseteq R_1[A_{11}] )</td>
</tr>
<tr>
<td>( R_3(A_{31}) )</td>
<td>( A_{31} )</td>
<td>( R_6[A_{6}] \subseteq R_1[A_{11}] )</td>
</tr>
<tr>
<td>( R_4(A_{41}, A_{42}) )</td>
<td>( A_{42} )</td>
<td>( R_7[A_{7}] \subseteq R_5[A_{5}] )</td>
</tr>
<tr>
<td>( R_8(A_{61}, A_{62}) )</td>
<td>( A_{62} )</td>
<td>( R_8[A_{8}] \subseteq R_3[A_{3}] )</td>
</tr>
<tr>
<td>( R_7(A_{71}, A_{72}, A_{81}, A_{82}) )</td>
<td>( A_{72} )</td>
<td>( R_7[A_{7}] \subseteq R_4[A_{4}] )</td>
</tr>
<tr>
<td>( R_7[A_{7}, A_{91}, A_{92}] )</td>
<td>( A_{92} )</td>
<td>( R_7[A_{9}] \subseteq R_3[A_{3}] )</td>
</tr>
</tbody>
</table>

*Note:* *nulls are allowed for \( A_{81} \) and \( A_{91} \).*

---

Fig. 3 Relational Schema of Fig. 2 after Merging.
VI. REFERENTIAL INTEGRITY IN RELATIONAL DATABASE MANAGEMENT SYSTEMS

Referential integrity is currently supported by several relational database management systems (RDBMS), notably IBM's DB2 [7], SYBASE [18], and INGRES [8]. The referential integrity capabilities of these three RDBMSs are briefly examined below. A more detailed analysis is provided in [13].

6.1 IBM's DB2.

In DB2 referential integrity constraints are specified non-procedurally (declaratively), but with certain restrictions. We examine below these restrictions and their effect on the relational representation of object-oriented structures.

Let \( RS = (R, F \cup I) \) be a relational schema, where \( R \), \( F \), and \( I \) consist of relation-schemes, key dependencies, and referential integrity constraints, respectively; let \( G_I \) be the referential integrity graph associated with \( RS \). The referential integrity constraints of \( I \) must satisfy the following restrictions in DB2:

1. Every referential integrity constraint of \( I \) is considered to be associated (by default) with a restricted update-rule.

2. Let \( I' \) be a subset of \( I \) that consists of referential integrity constraints corresponding to edges forming a directed cycle in \( G_I \). If \( I' \) consists of a single constraint, then this constraint must be associated with a cascades delete-rule. If \( I' \) consists of two or more constraints, then at least two constraints of \( I' \) must be associated with restricted or nullifies delete-rules.

3. Let \( Casc(R_i) \) be defined as in proposition 2 of section 4. For every pair of vertices of \( R \), \( R_i \) and \( R_j \), if \( R_j \) is connected in \( G_I \) by multiple edges to vertices of \( (Casc(R_i) \cup \{R_i\}) \), then the referential integrity constraints corresponding to these edges must be associated with identical, restricted or cascades delete-rules. ■

Note that DB2 allows the specification of partly null foreign-keys and overlapping keys. We contrast below the DB2 restrictions above with the conditions satisfied by the referential integrity constraints of relational schemas representing EER object structures:

(i) DB2 does not support the cascades update-rules involved in the relational schemas representing EER object structures.

(ii) Restriction (2) above treats cycles consisting of single edges differently from cycles consisting of multiple edges. This apparent contradiction does not exist for the referential integrity constraints of relational schemas representing EER object structures (see condition (c) of proposition 3).

(iii) Restrictions (2) and (3) above are stronger than the conditions of propositions 1, 2, and 3. First, note that if every relation-scheme corresponds to a unique object-set, then restrictions (2) and (3) above are trivially satisfied. However, if relation-schemes are allowed to correspond to multiple object-sets (e.g. following mergings), then restrictions (2) and (3) above might not be satisfied. Consider, for example, the relational schemas of figures 4(ii) and 4(iv), generated by \( Rmap \) and \( Rmerge \), from the EER schemas of figures 4(i) and 4(iii), respectively. If the two existence dependencies of \( SUPERVISE \) on \( EMPLOYEE \) in the EER schema of figure 4(i) are both of type block (resp. trigger), then the referential integrity constraint in the relational schema of figure 4(ii) is associated with a restricted (resp. nullifies) delete-rule; consequently this schema does not satisfy restriction (2) above, while it satisfies the condition (c) of proposition 3.

(iv) Conversely, the referential integrity constraints of the relational schema of figure 4(iv) satisfy restriction (2) of the definition above, only if all the existence dependencies in the EER schema of figure 4(iii) are of type block; however, if the existence dependencies are of type trigger, then \( R_1[A_1] \subseteq R_2[A_2] \) is associated with a nullifies delete-rule, while \( R_2[A_2] \subseteq R_1[A_1] \) is associated with a cascades delete-rule; therefore, restriction (2) above is not satisfied, while condition (c) of proposition 3 is satisfied.

\[\text{Fig. 4 Relational Schema Examples.}\]

Notes: All relational attributes correspond to SSN.

* Nulls are allowed.

\[\text{Nulls are allowed.}\]
6.2 SYBASE and INGRES.

SYBASE [18] and INGRES [8] do not allow declarative specifications of referential integrity constraints. Instead, they provide mechanisms for specifying procedurally such constraints. We examine below the main characteristics and limitations of these mechanisms.

The mechanism provided by SYBASE for implementing referential integrity constraints involves triggers. Triggers are a special kind of stored procedures that are activated (fired) when a relation is affected by a data manipulation (i.e., an insertion, deletion, or update). A trigger procedure is associated with a unique relation, say \( r_i \), and employs two system provided relations, called deleted and inserted: the deleted relation consists of tuples of \( r_i \) that are going to be deleted or updated; the inserted relation consists of tuples that are going to be inserted into \( r_i \), or newly updated tuples of \( r_i \). SYBASE allows the specification of three trigger procedures per relation: an insert, a delete, and an update trigger procedure. Given relation \( r_i \) associated with relation-scheme \( R_i \), the trigger procedures for \( r_i \) are executed in order to enforce the referential integrity constraints involving \( R_i \). SYBASE imposes certain (rather arbitrary) technical limitations, such as the number of levels allowed for nesting triggers. Another limitation concerns the implementation of referential integrity constraints associated with cascades update-rules: a cascades update-rule can be implemented only if updates of primary-key attributes are restricted to single tuples at a time, that is, only if the inserted and deleted relations consist of single tuples.

The INGRES rule mechanism [8] is conceptually similar to the SYBASE trigger mechanism. The differences, which are mainly of a technical nature, are summarized below:

(i) Unlike SYBASE, INGRES does not restrict the number of triggers per relation, nor the depth of rule nesting.

(ii) While SYBASE triggers are set-oriented (i.e. are activated for sets of tuple manipulations), INGRES rules are tuple-oriented (i.e. are activated for single tuple manipulations); accordingly, INGRES rules present no problem in implementing cascades update-rules.

(iii) INGRES provides a more evolved mechanism for handling errors and messages, and INGRES's Embedded SQL employed for specifying rules is more flexible than SYBASE's Transact-SQL employed for specifying triggers.

Overall, the INGRES rule mechanism is technically superior to the SYBASE trigger mechanism. However, both the SYBASE trigger and the INGRES rule mechanisms are extremely cumbersome. The use of SQL, compounded in SYBASE by certain syntactic limitations, make trigger and rule procedures very large and hard to comprehend. The manual specification of trigger and rule procedures is a tedious and error-prone process that tends to discourage users from specifying them for non-trivial databases.

VII. SUMMARY.

We have examined the concept of referential integrity in the context of relational schemas representing EER object structures. We have explored the effect of different relational representations of EER schemas on the structure of referential integrity constraints. Our analysis shows that the referential integrity concept can be simplified, and that the controversial referential integrity structures can be avoided without affecting the capability of relational schemas to represent EER object structures.

We have discussed the referential integrity capabilities of three relational database management systems (RDBMS), DB2, SYBASE, and INGRES. We have shown that some restrictions imposed by DB2 on the structure of referential integrity constraints limit the capability of defining in DB2 relational schemas representing EER object structures. We have compared the mechanisms provided by SYBASE and INGRES for the procedural specification of referential integrity constraints. We have shown that although conceptually similar, these mechanisms have technical differences, with the INGRES rule mechanism being more flexible and less restrictive than the SYBASE trigger mechanism.

Finally, we have pointed out the difficulty of using SYBASE triggers and INGRES rules. Using these mechanisms is labor-intensive and error-prone, therefore users should be insulated from them by a high-level interface that would allow non-procedural specifications of referential integrity constraints, and that would detect erroneous referential integrity structures. We have implemented such an interface as part of a Schema Design and Translation (SDT) tool [17]. SDT generates (i) abstract relational schemas from EER schemas, and (ii) RDBMS schema definitions from abstract relational schemas. Currently, SDT targets DB2, SYBASE, and INGRES. We have employed SDT for the generation of SYBASE and INGRES schema definitions from EER schemas consisting of approximately twenty object-sets. The difficulty of specifying SYBASE triggers and INGRES rules is illustrated by the
amount of code (over one thousand lines) generated by 

SDT for the trigger and rule procedures involved in 

these definitions.

REFERENCES

[1] P.P. Chen, "The entity-relationship model-
towards a unified view of data", ACM TODS 1, 1 
(March 1976), pp. 9-36.
[2] E.F. Codd, "Extending the relational database model to capture more meaning", ACM TODS 4, 
4 (Dec 1979), pp. 397-434.
Record 15,4 (Dec 1986), pp. 53-77.
[4] S.S. Cosmadakis and P.C. Kanellakis, "Equational Theories and Database Constraints", in Proc. of 
[5] C.J. Date, "Referential integrity", in Relational Database-Selected Writings, Addison-Wesley, 
1986.
[6] C.J. Date, "Referential integrity and foreign keys: Further considerations", in Relational Database- 
ACM TODS 9, 2 (June 1984), pp. 283-308.
1990.
1990.

APPENDIX

In this appendix we present two procedures, for mapping EER schemas into relational schemas, and for merging relation-schemes in relational schemas representing EER schemas.

A.1 Mapping EER Schemas.

The procedure for mapping EER schemas into relational schemas is based on procedures developed in [15]. We refer below to generalization-sources, which denote entity-sets that are not specializations of other entity-sets, and independent entity-sets, which denote generalization-sources that are not weak entity-sets. We assume that EER schemas satisfy certain well-definedness properties, such as the acyclicity of EER diagrams and the uniqueness of generalization-sources for specializations; these properties are discussed in [15] (see also [9] for a related discussion).

Definition – Rmap.

Input: a well-defined EER schema;

Output: a relational schema of the form \((R, F, I)\).

1. Value-Sets. Every value-set is mapped into a relational domain.

2. Independent Entity-Sets. An independent entity-set \(E_i\) is mapped into a relation-scheme, \(R_i(X_i)\), such that: attribute set \(X_i\) is in a one-to-one correspondence with the EER attributes of \(E_i\); every attribute \(A\) of \(X_i\) is assigned the domain corresponding to the value-set of the EER attribute of \(E_i\) corresponding to \(A\). The subset \(Z_i\) of \(X_i\) that is in a one-to-one correspondence with the identifier of \(E_i\) is specified
as the primary-key of $R_i$, and key dependency $R_i: X_i \rightarrow Y_i$ is added to $F$.

3. **Aggregation Object-Sets.** Let object-set $O_i$ be the aggregation of (not necessarily distinct) object-sets $O_{i,j}, 1 \leq j \leq m$, and let object-sets $O_{i,j}$ correspond to relation-scheme $R_{i}(Y_{i,j}), 1 \leq j \leq m$, respectively. Object-set $O_i$ is mapped into relation-scheme $R_i(X_i)$, and inclusion dependencies $R_i[FK_{i}] \subseteq R_i[K_{i}], 1 \leq j \leq m$, are added to $I$. Attribute set $X_i$ is the union of two disjoint sets of attributes, $X_i'$ and $X_i''$, such that: (a) $X_i'$ is in a one-to-one correspondence with the EER attributes of $O_i$, where the correspondence is specified as in (2) above; (b) $X_i'' = \bigcup_{j=1}^{m} FK_{i,j}$, is a set of foreign-key attributes, where every foreign-key $FK_{i,j}$ is in a one-to-one correspondence with primary-key $K_{i,j}, 1 \leq j \leq m$, such that every attribute $A$ of $FK_{i,j}$ is assigned the domain associated with the attribute of $K_{i,j}$ corresponding to $A$.

If $O_i$ is a *weak entity-set* and $Z_i$ is the subset of $X_i$ that is in a one-to-one correspondence with the identifier of $E_i$, then $Z_i X_i''$ is specified as the primary-key of $R_i$, and key dependency $R_i: Z_i X_i'' \rightarrow X_i$ is added to $F$. If $O_i$ is a *relationship-set* then if all the cardinalities of the object-sets involved in $O_i$ are *many*, then $X_i''$ is specified as the primary-key of $R_i$, and key dependency $R_i: X_i'' \rightarrow X_i$ is added to $F$; else ($X_i'' - FK_{i}$) is specified as the primary-key of $R_i$, where $FK_{i}$ is the *foreign-key* referencing the relation-scheme corresponding to an object-set that has cardinality *one* in $O_i$, and for every object-set $O_{i,j}$ which has cardinality *one* in $O_i$, key dependency $R_i: (X_i'' - FK_{i}) \rightarrow FK_{i,j}$ is added to $F$.

4. **Specialization Entity-Sets.** Let entity-set $E_i$ be the specialization of entity-sets $E_{i,j}, 1 \leq j \leq m$, and $E_b$ be the (unique) generalization-source of $E_i$. Let $E_i$ correspond to relation-scheme $R_b(Y_b)$ and entity-sets $E_{i,j}$ correspond to relation-schemes $R_{i,j}(Y_{i,j}), 1 \leq j \leq m$, respectively. Entity-set $E_i$ is mapped into relation-scheme $R_i(X_i)$, and inclusion dependencies $R_i[FK_{i}] \subseteq R_i[K_{i}], 1 \leq j \leq m$, are added to $I$. Attribute set $X_i$ is the union of two disjoint sets of attributes, $X_i'$ and $X_i''$, such that: (a) $X_i'$ is in a one-to-one correspondence with the EER attributes of $E_i$, where the correspondence is specified as in (2) above; (b) $X_i''$ is in a one-to-one correspondence with the primary-key of $R_i$, where the correspondence is specified as in (3.b) above; and (c) every foreign-key $FK_{i,j}, 1 \leq j \leq m$, is equal to $X_i''$. $X_i''$ is specified as the primary-key of $R_i$, and key dependency $R_i: X_i'' \rightarrow X_i$ is added to $F$. ■

A.2 Merging Relations.

The merging procedure below is a restricted version of the procedure developed in [12]. In the definition of this procedure we use the relational algebra operations of total projection, renaming, and outer equi-join [10].

Let $R_i(X_i)$ be a relation-scheme associated with relation $r_i$, and $W$ be a subset of $X_i$. The total projection of $r_i$ on $W$ is denoted $\pi_W(r_i)$, and is equal to \{ $t[W] \mid t \in r_i \text{ and } t \text{ consists of non-null values} \}$. Let $R_i(X_i)$, $r_i$, and $W$ be defined as above, and let $Y$ be an attribute set compatible with $W$. Renaming $W$ to $Y$ in $r_i$ is denoted $\text{rename}(r_i,W \mapsto Y)$, and is equal to \{ $t'[Y] = t[W] \mid t \in r_i \text{ and } t'[Y] = t[W]$ \}. Let $R_i(X_i)$ and $R_j(X_j)$ be two relation-schemes associated with relations $r_i$ and $r_j$, respectively; let $Y$ and $Z$ be two compatible and disjoint subsets of $X_i$ and $X_j$, respectively; let $k_i$ and $k_j$ denote the number of attributes in $X_i$ and $X_j$, respectively. We denote by $\omega$ a null value, and by $\omega_k$ a tuple consisting of $k$ null values. The *outer-equijoin* of $r_i$ and $r_j$ on $(Y=Z)$ is denoted $r_i \bowtie_{Y=Z} r_j$, and is equal to the union of three relations: $r_1 = \{ t \mid t[X_i] \in r_i, t[X_j] \in r_j, t[Y] = t[Z] \};$ $r_2 = \{ t \mid t[X_i] = \omega_i, t[X_j] \in r_j, t[Y] = t[Z] \}$; and $r_3 = \{ t \mid t[X_i] \in r_i, t[X_j] = \omega_j, t[Y] = t[Z] \}$; and $\not \exists t' \in r_i \text{ s.t. } t'[Y] = t[Z]$; and $\not \exists t'' \in r_j \text{ s.t. } t[Y] = t''[Z]$. The definition of *Merge*. 

**Input:** a relational schema $RS = (R, F \cup I)$, and a subset of $R$, $\bar{R}$, such that

(a) the primary-keys of the relation-schemes of $\bar{R}$ are pairwise compatible;

(b) there exists a relation-scheme $R_p(Y_p)$ in $\bar{R}$ such that for every $R_i$ of $\bar{R}$, $i \neq p$, $R_i$ is involved in an inclusion dependency of $I$ of the form $R_i[K_i] \subseteq R_p[K_p] \in I$;

(c) every relation-scheme $R_i(X_i)$ of $\bar{R}, i \neq p$:

(i) has exactly one non primary-key attribute;

(ii) cannot be involved in the right-hand side of any inclusion dependency of $I$; and

(iii) can be involved in the left-hand side of at most two inclusion dependencies, one involving $R_p$ (see (b) above), and one of the form $R_i[X_i-K_i] \subseteq R_j[K_j]$.

**Output:** a relational schema $RS' = (R', F' \cup I')$. 

-11-
**Rmerge** \((\bar{R})\) applied on \(RS\) generates \(RS'\) as follows:

1. \(R'\) results by replacing the relation-schemes of \(\bar{R}\) in \(R\) with relation-scheme \(R_m(X_m)\), such that \(K_m := K_p, \ X_m := K_m \cup \bigcup_{R_i \in \bar{R}} (X_i - K_i)\), where the attributes of \((X_m - X_p)\) are allowed to have null values;

2. \(F'\) results by replacing in \(F\) all the key dependencies involving primary-keys associated with the relation-schemes of \(\bar{R}\), with key dependency \(R_m : K_m \rightarrow X_m\);

3. \(I'\) results by replacing \(R_i\) with \(R_m\) and \(K_i\) with \(K_m\), in every inclusion dependency of \(I\) that involves a relation-scheme \(R_i\) of \(\bar{R}\).

**Rmerge** \((\bar{R})\) is associated with two state mappings, \(\eta\) and \(\eta'\), where \(\eta\) maps a state \(r\) of \(RS\) into a state \(r'\) of \(RS'\), and \(\eta'\) maps a state \(r'\) of \(RS'\) into a state \(\bar{r}\) of \(RS\), as follows:

\(\eta\) is the identity for the relations of \(r\) that are associated with relation-schemes of \((\bar{R} - \bar{R})\); and maps the set of relations \(\{r_i | r_i \in r, r_i\ \text{is} \ \text{associated} \ \text{with} \ \bar{R}_i \in \bar{R}\}\) into \(r_m\) as follows:

(i) \(r_m := r_p;\) and

(ii) for each \(R_i\) of \((\bar{R} - \{R_p\})\)

\[
\text{do } \ r_m := \pi_{(X_m - K_m)}(r_m \odot \bigcup \{r_i\}) \text{ enddo.}
\]

\(\eta'\) is the identity for every relation of \((r' - \{r'_m\})\); and maps relation \(r'_m\) of \(r'\) into relations \(\bar{r}_i\) as follows:

\[
\bar{r}_i := \text{rename} (\pi_{X_m(K_m - K_i)} (r'_m, K_m \leftarrow K_i),
\]

where \(R_i(X_i)\) is a relation-scheme of \(\bar{R}\). 

-12-